dc_244_11

PRECÍZ MODELLTRANSZFORMÁCIÓK TERVEZÉSE ÉS ANALÍZISE A MODELLVEZÉRELT FEJLESZTÉSBEN

MTA DOKTORI ÉRTEKEZÉS TÉZISEI

VARRÓ DÁNIEL

BUDAPEST, 2011

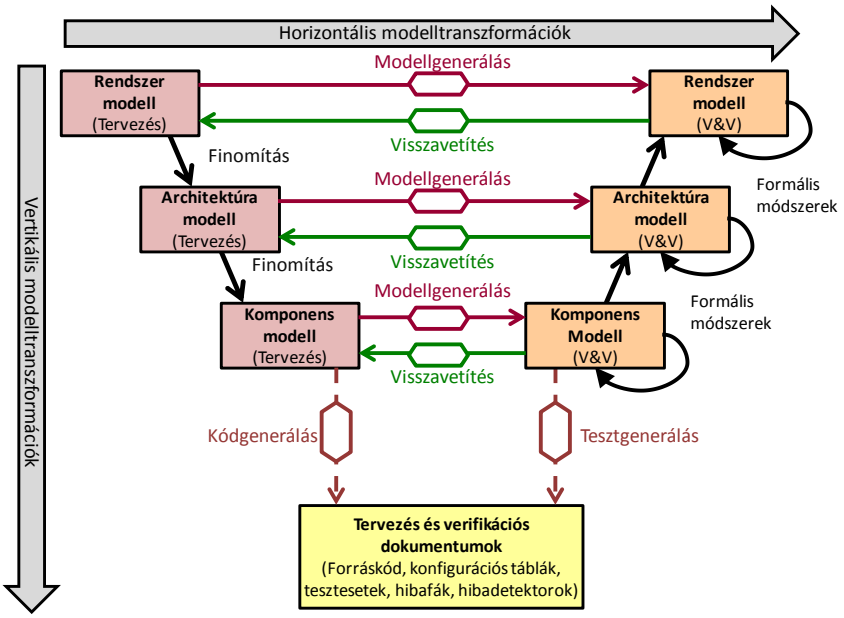
Tartalomjegyzék

1. Bevezetés

1.1. Modellvezérelt fejlesztés a szolgáltatásbiztos rendszertervezésben

Napjainkban teljes vállalatok működése függ az üzletmenet szempontjából kritikus szolgáltatásoktól, amelyek hibás működése komoly pénzügyi veszteségeket okozhat. Mi több, e szolgáltatásoknak adaptívnak és robosztusnak kell lenniük változó üzleti környezetben is. Biztonságkritikus rendszerekben egy kritikus komponens hibája akár emberéletben mért veszteséget is okozhat. E területen a tanúsítványozási szabványok által megkövetelt módon, matematikailag precíz verifikációs és validációs módszerekkel kell demonstrálni, hogy a rendszer mentes a tervezési és implementációs hibáktól. A *szolgáltatásbiztos rendszerek tervezése* tehát jelenleg is a szoftvertervezés egyik legnagyobb kihívása.

A **modellvezérelt fejlesztés** [21, 39] napjainkban kulcstechnológia a szolgáltatásbiztos rendszertervezés területén (1. ábra), amely különböző absztrakciós szintű modellek szisztematikus használatát írja elő már a tervezési folyamat legkorábbi fázisaitól kezdve egészen a már működésben lévő rendszer vezérléséig [36]. A rendszermodellek létrehozására célszerű valamilyen *magasszintű, szabványos grafikus tervezőnyelv* használata (például UML, SysML, AADL vagy BPMN). A rendszermodellből *automatikus modelltranszformációk* segítségével (horizontális transzformációk) matematikai modelleket generálhatunk, így elvégezhető a rendszermodellek *formális módszerekkel támogatott korai analízise* [3, 30]. A matematikai analízis eredményeinek visszavetítése után a rendszermérnök a tervezési hibák, inkonzisztenciák listáját kapja kézhez, amelyet még a rendszermodell szintjén javíthat. A rendszermodell számos további optimalizálási lépés alapjául is szolgálhat [17, 24]. Végezetül az optimális és formálisan helyes rendszermodellekből *automatikus kódgenerálással* (vertikális transzformációk) származtathatjuk a szoftverrendszer forráskódját, a telepítési leírókat, teszteseteket stb.



1. ábra. Kritikus rendszerek modellvezérelt tervezése

1.2. A modellvezérelt fejlesztés előnyei

Nemrégiben készült felmérések alapján elmondható, hogy az automatikus kódgenerátorok (mint a repülőgépiparban használt SCADE, vagy az üzleti folyamatok modellezésére használt IBM Websphere Business Modeler) által támogatott modellvezérelt tervezés szignifikánsan növeli a mérnökök produktivitását és a szoftverrendszer minőségét. A repüléstechnikai szoftverek tervezésekor a fejlesztési költségek közel 50%-kal, a tesztelési költségek pedig mintegy 30%-kal csökkenhetnek a biztonságos, tanúsítványozási szabványokkal (pl. DO-178B Level A) kompatibilis forráskód automatikus szintézisével [25,30]. A kódolási hibák kiiktatása és a nyomonkövethetőség szisztematikus biztosítása felére csökkenti a tanúsítványozás összköltségét. Hasonló megtakarításokat jelentettek precízen modellezett üzleti folyamatok és szolgáltatás-orientált rendszerek esetén is.

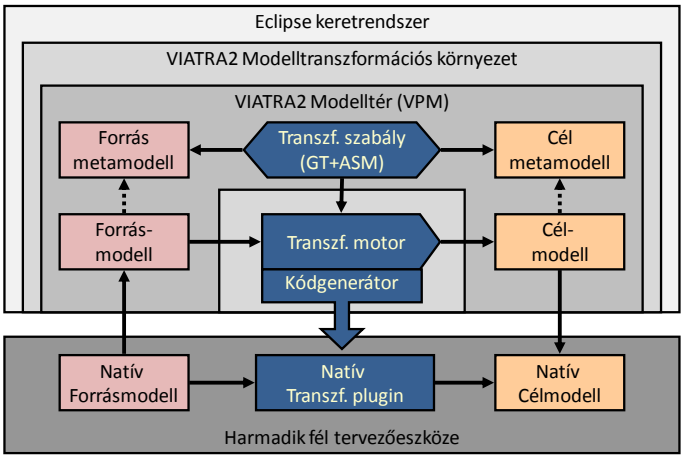
Az automatikus modelltranszformációk elsődleges célja, hogy hatékonyan ültesse át a formális módszerek és automatikus kódgenerálás nyújtotta elméleti eredményeket a modern szoftver- és rendszertervezés gyakorlatába, automatizálva a mérnöki modellek és matematikai modellek közötti átjárást [5, 34]. Az elmúlt években számos nemzetközi kutatási projekt (úgy mint TOPCASED, DECOS, ASSERT vagy DIANA a biztonságkritikus rendszerek területén, illetve DEGAS, DEPLOY vagy SENSORIA a szolgáltatás-orientált rendszerek területén) vizsgálta az automatikus modelltranszformációk hasznosítását.

Koncepcionálisan a modelltranszformáció egy (automatikus vagy felhasználó által vezérelt) fordítási lépésnek tekinthető, amely egy (vagy több) forrásmodellből egy (vagy több) célmodellt állít elő. A hagyományos fordítóprogramok elméletéhez képest ugyanakkor itt nemcsak a szöveges nyelvekre jellemző fastruktúrák feletti, hanem grafikus modellező nyelvekre jellemző komplex gráfstruktúrák feletti leképezéseket is meg kell valósítanunk. Továbbá maguk a transzformációs lépések is gyakorta adatvezéreltek, szemben a hagyományos fordítók vezérlés-orientált szemléletével.

Számos akadémiai és ipari alkalmazás során bebizonyosodott, hogy az automatikus modelltranszformációk kiemelten jó minősége elengedhetetlen a modellvezérelt tervezés sikeres alkalmazásában, ezért precíz szoftvertervezési technikákra van szükség a modelltranszformációk fejlesztésének teljes életciklusa folyamán, beleértve a specifikáció, tervezés, végrehajtás és validálás fázisait is.

1.3. Modelltranszformációk fogalmi áttekintése

A modellezési nyelvek közötti transzformációk főbb fogalmait a 2. ábra összegzi. Első lépésként a transzformációk forrás- és a célnyelvét kell definiálni a *metamodelljeik* által. Tervezéskor a modelltranszformációt a *transzformációs szabályok* egy halmazával definiálhatjuk. Végrehajtáskor, azaz egy *modelltranszformációs futáskor* egy modelltranszformációs *futtatómotor értelmezi és végrehajtja a transzformációs szabályokat*, hogy egy bemeneti *forrásmodellből* (a forrásnyelv egy példányából) egy *célmodellt* (a célnyelv egy példányát) generáljon. Mivel a forrás- és célmodellek legtöbbször valamilyen natív formátumban adottak, e modellek belső gráfreprezentációját *importerek* és *exporterek* segítségével állítjuk elő. Alternatív megközelítésként *modelltranszformációs programok* (más néven pluginek) generálhatók egy transzformációs szabályhalmazból [K2, K26] (3. tézis), amelyek azután beágyazhatók egy harmadik fél által gyártott tervezőeszközbe.



2. ábra. A modelltranszformáció alapfogalmai

2. Problémafelvetés és kutatási célok

2.1. Modelltranszformációk specifikációja

Noha számos modelltranszformációs eszköz áll rendelkezésre, a valós ipari gyakorlatban a legtöbb modelltranszformációt még mindig ad hoc módon, kézzel írják, éppúgy mint egy hagyományos szoftverprogramot. Ez részben annak köszönhető, hogy a rendelkezésre álló modelltranszformációs specifikációs nyelvek vagy matematikailag precízek, de nem intuitívak (amely nehezíti a széleskörű ipari alkalmazásukat), vagy intuitívak, de nem precízek (amely gátolja az alkalmazásukat kritikus rendszerek tervezésekor). Sajnos a népszerű fejlesztőkörnyezetekkel ellentétben a modelltranszformációs keretrendszerek ritkán tartalmaznak generikus, újrafelhasználható transzformációs komponenseket.

A modelltranszformációs nyelvek absztrakciós szintje fokozatos növekekszik. A deklaratív, szabályalapú nyelvek közé tartozik a QVT (Queries, Views and Transformations) [28] szabványos modelltranszformációs nyelve, amely azonban — precíz formális alapok híján — több, egymással inkompatibilis végrehajtási szemantikával is rendelkezik [14], így ipari használatra alkalmatlan. Kutatásom első kihívása így fogalmazható meg:

1. kihívás (Modelltranszformációk specifikációja). *A modelltranszformációk egy matematikailag precíz, ugyanakkor intuitív specifikációs nyelvet tesznek szükségessé, amely támogatja a generikus, újrafelhasználható transzformációs komponensek létrehozását.*

2.2. Modelltranszformációk tervezése

A modellezési nyelvek közötti automatikus modelltranszformációk tervezése a modellvezérelt fejlesztés központi problémájává vált. Számos nagy kifejezőerővel rendelkező transzformációs nyelv fejlődött ki az elmúlt években.

Ugyanakkor közös probléma, hogy e transzformációs nyelvek szignifikánsan különböznek a transzformált forrás- és célmodellek nyelvétől. Ennek következtében a transzformációk tervezőinek nemcsak a transzformációs problémát kell megérteniük (azaz hogyan képezzük le a forrásmodelleket célmodellekre), hanem jelentős mélységben ismerniük

kell magát a transzformációs nyelvet is, hogy megfogalmazhassák a megoldást. Sajnos a legtöbb szakterületi szakértő (akik a forrás- és célnyelv ismerői) ritkán sajátítja el ilyen mélységben a modelltranszformációs technológiákat, így a kutatásom második kihívása így fogalmazható meg:

2. kihívás (Modelltranszformációk tervezése). *A modelltranszformációk tervezésére olyan technikák szükségesek, amelyek lehetővé teszik a szakterületi szakértők számára a modelltranszformációk fejlesztését anélkül, hogy jártasak lennének a konkrét modelltranszformációs nyelvekben.*

2.3. Modelltranszformációk végrehajtása

Komplex ipari alkalmazások esetén a modelltranszformációknak hatalmas méretű modellek felett kell lekérdezéseket és modellmanipulációkat végezni, amelyek több százezer vagy akár millió modellelemből állnak (mint például az AUTOSAR modellek az autóipar területén vagy a SysML és AADL modellek a repülésiparban).

Sajnos a legtöbb modelltranszformációs eszköz képtelen hasonló méretű modellek kezelésére, amely jelentősen limitálja a felhasználási területüket. Többek között az iparban népszerű Eclipse M2M keretrendszer modelltranszformációs eszközeinek skálázhatósága is problematikus 20-50 ezer modellelem fölött.

A futtatási teljesítmény problémáján túl a modelltranszformációs szabályok legtöbbször egy komplex transzformációs keretrendszeren belül hajthatók csak végre, amely nem teszi lehetővé, hogy egyes modelltranszformációkat egy harmadik fél fejlesztőeszközébe integráljunk. Például ha egy modelltranszformáció két adatbázisban tárolt (és adatbázis séma által definiált) modell között hajt végre transzformációt, akkor célszerű lenne magát a transzformációt is az adatbázis technológiába ágyazni (pl. SQL szkriptek előállításával vagy natív Java programok formájában). A fentiek miatt a harmadik kihívás a következőképpen foglalható össze:

3. kihívás. *A modelltranszformációk hatékony, nagyméretű modelleket kezelni képes végrehajtási technikákat tesznek szükségessé, amelyek jól integrálhatók nyílt vagy harmadik fél által készített fejlesztőeszközökbe.*

2.4. Modelltranszformációk helyessége

Napjainkban számos nyelv és tervezőeszköz áll a szoftvermérnökök rendelkezésére a modelltranszformációk automatikus végrehajtására (például QVT [28], ATL [18]). Sajnos azonban az automatikus modelltranszformációk is lehetnek hibásak, egy hibás transzformáció pedig érvényteleníthet a precíz matematikai analízis eredményét [19, K25]. Ezért garantálnunk kell, hogy maguk a transzformációk is mentesek a tervezési hibáktól, azaz igazolnunk kell, hogy a célmodellben detektált tervezési hibákért semmi esetre sem a modelltranszformáció a felelős. Ellenkező esetben nem tudnánk különbséget tenni, hogy egy célmodell hibája már a forrásmodellben is jelen volt, vagy csak a (hibás) transzformáció folyamán került be a rendszerbe. A transzformációs hibák — a fordítóprogramok hibáihoz hasonlóan — nehezen és költségesen detektálhatók, tanúsítványozott (azaz garantáltan helyes működésű) modelltranszformációs gép pedig nem ismert.

Sajnos a modelltranszformációk verifikációját és validációját célzó kutatások igen korai fázisban vannak. Egyrészt ad hoc problémákat céloznak meg, másrészt a kidolgozott technikák skálázhatósága is kérdéses. Szintén problémát jelent, hogy a létező validációs technikák nem illeszkednek a kritikus rendszerek tervezése során használt szabványos tanúsítványozási folyamatok által előírt követelményekhez és folyamatokhoz. Ezért egy transzformáció által generált célmodellt úgy kell kezelni, mintha azt a tervező saját maga (kézzel) állította volna elő, és így igen költséges újravalidáció szükséges.

4. kihívás. *A modelltranszformációk helyességének formális ellenőrzésére matematikai precizitású verifikációs és validációs technikák kidolgozása szükséges.*

Helyességi kritériumok Számos helyességi kritérium definiálható a modelltranszformációk kontextusában [K25]. Egy minimális követelmény a *szintaktikus helyesség* biztosítása, amely garantálja, hogy a generált célmodell a célnyelv szintaktikusan jólformált példánya. A *szintaktikus teljesség* megköveteli, hogy a forrásnyelvet teljesen lefedjük transzformációs szabályokkal, azaz a forrásnyelv minden egyes eleméhez létezzen egy megfelelő elem a célmodellben. Kiemelt minőségű modelltranszformációk esetén azonban a következő *szemantikai követelményeket* is vizsgálnunk kell.

- **Terminálás:** Egy alapvető követelmény a modelltranszformációk terminálásának vizsgálata. Ez a modelltranszformációk helyességének egy általános, nyelvfüggetlen szemantikus kritériuma, amely általános esetben eldönthetetlen [31].
- **Determinisztikusság (Egyértelműség, konfluencia):** Mivel a gráftranszformáció alapú modelltranszformációs specifikációk természetes velejárója az egyes szabályok nemdeterminizmusa (például melyik illeszkedés mentén hajtsunk végre egy szabályt), garantálnunk kell, hogy a transzformáció egésze determinisztikus legyen, azaz minden forrásmodellre egyértelmű végeredményt adjon. Ez szintén egy nyelvfüggetlen helyességi kritérium.
- **Szemantikus helyesség (tulajdonságmegőrzés):** Elméletben egy kézenfekvő helyességi kritériumnak tűnhet a forrás- és célmodellek ekvivalenciájának bizonyítása. A modelltranszformációk azonban igen gyakran egy projekciót definiálnak a forrásnyelvről a célnyelvre (melynek során szándékoltan információt veszítünk); ilyenkor a szemantikus ekvivalencia nem igazolható. Célszerűbb tehát olyan (transzformáció-specifikus) *helyességi kritériumot* felállítani, *amelyek megőrzését megköveteljük egy szemantikailag helyes transzformációtól.*

A transzformációk szintaktikus és szemantikus helyességének kérdését vizsgálja többek között [1,23,B5,J12,J13,K22,K25]. A jelen disszertációban a *modelltranszformációk terminálásának* kérdését tárgyalom.

2.5. Célkitűzés

Értekezésemben új módszereket dolgoztam ki a modelltranszformációk precíz specifikációjára, tervezésére, végrehajtására és formális analízisére. E technikák lehetővé teszik a modelltranszformációk szisztematikus szintézisét a szolgáltatásbiztos rendszerek modellvezérelt tervezése területén, automatizálva elsődlegesen a formális módszerekkel támogatott korai modellanalízis és a kódgenerálás lépéseit.

3. Új tudományos eredmények

3.1. Modelltranszformációk specifikációs technikái

Kutatási módszer Szolgáltatásbiztos rendszerek tervezésekor számos szakterület-specifikus modellező nyelvet használnak, amelyek — formális szemantikájuknak köszönhetően — elősegítik a rendszerspecifikáció egyértelműségét. E rendszerek tervezésekor hasonló formális precizitás szükséges a modelltranszformációk specifikációjára is.

Napjainkban a modelltranszformációk nagy többségét is szoftvermérnökök vagy szakterületi szakértők készítik, akik jártasak egy modelltranszformáció forrás- és a célnyelvének szemantikai részleteiben. Ugyanakkor, a programozási nyelvek túlnyomó részével ellentétben a modelltranszformációk tipikusan adatvezéreltek, melynek következtében leginkább deklaratív, szabálybázisú transzformációs formalizmusok alakultak ki. A procedurális vagy objektum-orientált programnyelvekkel összevetve azonban a szoftvermérnökök nehezebben sajátítják el egy tisztán deklaratív transzformációs nyelv használatát. Ezért a szolgáltatásbiztos rendszerek tervezése során használt *modelltranszformációk specifikációjára* olyan formalizmust célszerű kifejleszteni, amely *egyszerre matematikailag precíz és mérnökileg intuitív*.

Az újrafelhasználható, általános célú transzformációs komponensek létrehozása céljából két fő elképzelést vizsgáltam. Egyrészt a generikus modelltranszformációk (például tranzitív lezárt számítása) függetlenek az alkalmazásterülettől (azaz a transzformáció forrás és célnyelvétől), így a *transzformáció specifikációja során típusparamétereket vagy típusváltozókat használhatunk*. Ez hasonlatos az objektum-orientált programnyelvek (például Java vagy C++) makróihoz és generikus sablonjaihoz. Továbbá maguk egy *modelltranszformációk tárolható transzformációs modellként*, amely így további modelltranszformációk bemeneteként vagy kimeneteként szolgálhat.

Ezért *a modelltranszformációs nyelveknek kellően általánosnak* és nagy kifejezőerővel rendelkezőnek kell lenniük, hogy segítségükkel újrafelhasználható és jól adaptálható transzformációs komponenseket készíthessünk.

Megvizsgálva két népszerű, formális specifikációs módszert, nevezetesen a gráftranszformáció [9, 35] és az absztrakt állapotgépek [15] paradigmáját, azt találtam, hogy (1) a gráftranszformációs szabályok egy intuitív és deklaratív módszert adnak az elemi transzformációs lépések leírására, ugyanakkor komplex transzformációkat körülményes megadni kizárólag e tisztán deklaratív technika által. Ugyanakkor (2) az absztrakt állapotgépek segítségével [15] értelmében bármilyen komplex algoritmus leírható (beleértve a modellezési nyelvek dinamikus szemantikáját [6]), és korlátozható a gráftranszformációs szabályok nemdeterminizmusa is. Viszont a komplex gráf alapú modellstruktúrák algebrai leírása bonyolult lehet e formalizmusban.

Új tudományos eredmények Az 1. tézisben a gráftranszformáció és absztrakt állapotgépek kombinációjára épülő hibrid és generikus modelltranszformációs nyelvet dolgoztam ki, amely ötvözi a deklaratív és a procedurális paradigma előnyeit, továbbá támogatja az újrafelhasználható modelltranszformációs komponensek létrehozását.

1. tézis. Egy általános célú, formális specifikációs nyelvet javasoltam modellezési nyelveken belüli és modellezési nyelvek közötti modelltranszformációk definiálására [B3, B6, J5, J11, J14, K1, K6, K8, K10, K26, K28].

1/1 **Gráfminta alapú lekérdezőnyelv.** Modelltranszformációk körében lekérdező és kényszerleíró nyelvként bevezettem az általános rekurziót és tetszőleges mélységű negációt támogató gráfmintákat.

1/2 **Hibrid modelltranszformációs nyelv.** Az absztrakt állapotgépek és gráftranszformáció paradigmájának kombinálásával hibrid formális nyelvet javasoltam a modelltranszformációk precíz specifikációjára.

1/3 **Generikus transzformációk.** Magasabbrendű, generikus transzformációkat vezettem be különféle transzformációs problémák egységes leírására.

Az általam javasolt gráfminta alapú kényszerleíró- és lekérdezőnyelvre, valamint a hibrid modelltranszformációs nyelvre építve Varró Gergely és Horváth Ákos a rekurzív gráfminták egy hatékony mintaillesztési módszerét, Balogh András pedig a gráfminták kompozícióját és kódgenerálási sablonnyelvét dolgozta ki, mely PhD kutatásokban témavezetőként vagy társkonzulensként működtem közre. A generikus és meta-transzformációkat Pataricza András professzorral közösen dolgoztuk ki, ahol a személyes kontribúcióm elsődlegesen a generikus transzformáció fogalma volt.

Gyakorlati jelentőség Az absztrakt állapotgépek és gráftranszformáció paradigmájának kombinálásával létrehozott hibrid formális nyelv egy precíz, ugyanakkor közérthető formalizmust ad a modellezési nyelveken belüli és modellezési nyelvek közötti modelltranszformációk támogatására. E nyelv a VIATRA2 modelltranszformációs keretrendszer transzformációs nyelvévé vált, amelyet számos kutatási projektben (DECOS, DIANA, SENSORIA, SecureChange vagy MOGENTES) sikerrel használtak a kutatócsoportunkon kívül számos akadémiai műhelyben és ipari körökben is.

3.2. Modelltranszformációk tervezési technikái

Kutatási módszer Hogy támogassam a szakterületi szakértőket a modelltranszformációk tervezésében, új elképzeléseket vizsgáltam a szabályalapú tudásreprezentációk szintézise területén. Érdekesnek találtam, hogy számos „példa alapú” módszert javasoltak a szoftvertudomány különböző területein, melyek a szakértőkkel együttműködve prototipikus példákat használnak elsődleges bemenetként a tudásszintézis céljára.

Fejlett XML transzformációs eszközökben az XSLT szabályok automatikusan generálhatók, miután összekötöttünk egyszerű forrás és cél XML dokumentumokat (query-by-example) [10, 22, 29, 40]. A lekérdezés-példák-alapján megközelítés [41] egy olyan nyelvet javasol relációs adatok lekérdezéseinek megkonstruálására, amelyben példa táblázatokat kell feltöltenünk sorokkal és kényszerekkel. A programozás-példák-alapján (programming-by-example) megközelítésben [7, 32] a programozó (gyakran a végfelhasználó) pél-

daadatokon elvégzett műveletekkel demonstrálja az elvégzendő feladatot, amelyeket a számítógép rögzít, és általánosítani próbál.

A modelltranszformációk kontextusában egy példa alapú megközelítés fő célkitűzése *a modelltranszformációs szabályok automatikus generálása*. Ugyanakkor, ahogy más kutatási területeken javasolt példa-alapú megközelítések esetén is kitűnt, a teljes automatizálás ritkán lehetséges, ezért iteratív interakció szükséges a transzformáció tervezőjével.

Mivel a modellek és transzformációik könnyen reprezentálhatók egy logikai programozási nyelven, a figyelmem a logikai programozás alapú tudásszintézis módszerei felé fordult, különös tekintettel az induktív logikai programozás (ILP) [26] területére, amely matematikailag megalapozott módon képes Prolog programok következtetési szabályait származtatni egy pozitív és negatív példákból álló tudásbázisból kiindulva.

Új tudományos eredmények A 2. tézisben egy új, *példa alapú* megközelítést javasolok a modelltranszformációs szabályok automatikus származtatására.

2. tézis. Definiáltam a *modelltranszformáció-példák-alapján* (model transformation by example, MTBE) megközelítést [B2, J1, K23, K24], amely egy iteratív és félautomatikus módszer a modelltranszformációs szabályok generálására prototipikus, összekötött forrás- és célmodellekből kiindulva, amelyek a modelltranszformációk kritikus eseteit írják le deklaratív módon.

2/1 **Iteratív MTBE folyamat.** Egy félautomatikus, iteratív MTBE folyamatot javasoltam, amely a következő lépésekből áll: (1) a tervező prototipikus, összetartozó forrás- és célmodellpárokat hoz létre, (2) a transzformációs szabályokat automatikusan generáljuk, (3) a tervező finomítja a generált szabályokat, (4) a végső szabályhalmazt automatikusan végrehajtjuk.

2/2 **Új MTBE fogalmak: kontextus és összekötöttségi analízis.** Definiáltam a kontextus és összekötöttségi analízis fogalmát mind a forrás-, mind a célmodelleken, hogy meghatározható legyen azon modellelemek környezete, amelyek nélkülözhetetlenek egy adott transzformációs szabály bal és jobb oldalának létrehozásához.

2/3 **Automatizálás induktív logikai programozás által.** A modelltranszformációs szabályok automatikus származtatására az induktív logikai programozás paradigmájának felhasználását javasoltam.

Gyakorlati jelentőség Az MTBE megközelítés legfőbb előnye, hogy a transzformáció tervezőjének elég a forrás és cél modellezési nyelv fogalmait ismernie a modelltranszformációk specifikációja során, míg az implementáció, azaz a konkrét modelltranszformációs szabályok (fél-)automatikusan generálhatók. A transzformáció tervezője először súgást ad a forrás- és célmodellek *lehetséges* összeköttetésére az összeköttetési metamo-

dell formájában. Ezután a transzformációs szabályok tényleges kontextuális feltételeit a prototipikus forrás- és célmodell párokból származtatjuk.

Az általam bevezetett „modelltranszformáció-példák-alapján” megközelítés igen aktívan kutatott területté vált az elmúlt öt év során. Hét független kutatócsoport kezdett kutatásokat olyan alternatív megoldásokat javasolva, mint [8, 11, 12, 20, 33, 37, 38], melyek több mint 90 független hivatkozást adtak a kapcsolódó publikációinkra [J1, K23, K24].

3.3. Hatékony végrehajtási technikák modelltranszformációkhoz

Kutatási módszer A modelltranszformációk gyakorlati alkalmazásai során a forrás- és célmodellek mérete akár az egymillió modellelemet is meghaladhatja. Ahhoz, hogy ekkora méreteket is kezelni tudjunk, hatékony és jól megalapozott technikákra van szükség a modellek lekérdezésére és manipulálására.

A létező modelltranszformációs keretrendszerek (kutatásom kezdetén 2004-ben) (1) kizárólag a *batch végrehajtási módot* támogatták, amikor a forrásmodellekből a célmodellek előállítása a felhasználó explicit kezdeményezésére történik. A batch módú transzformációk azonban nehezen alkalmazhatók számos ipari transzformációs feladatra (például modellek inkrementális modellszinkronizációja tervezőeszközök integrációja, diszkrét, eseményvezérelt rendszerek szabályalapú modellszimulációja, stb).

Ezért kidolgoztam az inkrementális modelltranszformációk egy módszerét. Az inkrementális transzformációknál az egyik fő probléma annak behatárolása, hogy a forrásmodell változása esetén a transzformáció mely részeit kell újra végrehajtanunk. Mivel egy modelltranszformációs szabály tipikusan (a modellek méretéhez képest) viszonylag kevés forrásmodellelemre illeszkedik, a transzformációs szabályok illeszkedéseinek explicit tárolása egy ígéretes megközelítésnek tűnt.

Meglepő volt továbbá, hogy (2) a létező modelltranszformációs keretrendszerek a transzformációs szabályok végrehajtásakor *kizárólag metamodell szintű tudást használtak fel a modellek navigálására és lekérdezésére* a mintaillesztési lépés során. Egy modelltranszformációs szabály ugyanakkor többféle végrehajtási stratégia (ún. keresési tervek) szerint is alkalmazható, és a transzformálandó modell struktúrája is jelentősen befolyásolhatja egy keresési terv hatékonyságát. Például egy keresési tervnek már az első lépésben el kell döntenie, hogy honnan kezdődjön a keresés, kiszámítva az illeszkedésre jelölt modellelemek egy kezdeti halmazát a típus vagy tartalmazási hierarchia figyelembe vételével. Ezért kutatásaimban megvizsgáltam, hogy modellpéldány specifikus információk hogyan használhatók fel a lokális keresést használó transzformációs szabályok végrehajtása során.

Végezetül azt vettem észre, hogy (3) a modelltranszformációs eszközök egy zárt technológiát képeznek, így *az egyes modelltranszformációk nehezen integrálhatók létező tervezői keretrendszerekbe*. Ez különösen az egyes modelltranszformációk és transzformációs eszközök tanúsítványozása során jelent különös problémát. Ezért megvizsgáltam, hogy a modelltranszformációs keretrendszerek architektúrája miként módosítható, hogy tisztán kettéválaszthassuk a modelltranszformáció tervezési és végrehajtási fázisát.

Új tudományos eredmények A 3. tézisben egy új végrehajtási architektúrát és különböző hatékony végrehajtási stratégiákat dolgoztam ki a modelltranszformációkhoz.

3. tézis. Hatékony végrehajtási stratégiákat és architektúrát javasoltam gráfminták és gráftranszformációs szabályok által specifikált modelltranszformációkhoz [J2–J4, J9, J11, J15–J17, J17–J19, K2–K7, K19–K21, K26, K28].

3/1 **Modell-specifikus keresési tervek.** Bevezettem a modell-specifikus keresési tervek elvét, amely modellpéldány szintű statisztikák alapján a transzformálandó modell aktuális struktúráját kihasználva növeli a gráftranszformációs szabályok lokális keresésen alapuló végrehajtásának hatékonyságát.

3/2 **Inkrementális gráftranszformációk.** Bevezettem a inkrementális gráftranszformáció módszerét, amely tárolja a gráftranszformációs szabályok illeszkedéseit, és inkrementálisan karbantartja azokat a gráfmodell változása esetén.

3/3 **Lefordított modelltranszformációs pluginek.** Lefordított modelltranszformációs programokra épülő egységes architektúrát definiáltam, amely elősegíti modelltranszformációs szabályok beágyazását natív célkörnyezetekbe.

E kutatásokat szoros együttműködésben végeztem több doktorandusz hallgatóval. Ráth István, Horváth Ákos és Bergmann Gábor esetén témavezetőként, Balogh András és Varró Gergely esetén társkonzulensként működtem közre. A saját kontribúcióm *a hatékony modelltranszformációk általános, magasszintű stratégiáinak és elveinek kidolgozása volt*, amelyek részletekbe menő kidolgozását a fenti doktoranduszok végezték.

Gyakorlati jelentőség Ezek az eredmények implementálásra kerültek a VIATRA2 modelltranszformációs keretrendszerben és az EMF-IncQuery inkrementális modell-lekérdezéseket támogató rendszerben [K5, K6, K8]. Több benchmark vizsgálatot is elvégeztünk [J2, J11, K3, K6], amelyek egyértelműen kimutatták, hogy a fenti elveken alapuló modelltranszformációk jól skálázódnak több millió modellelemből álló modellekre is.

Az inkrementális modelltranszformációk elvét általánosítva Bergmann Gáborral, Ráth Istvánnal, és Varró Gergellyel közösen nemrégiben bevezettük a *változásvezérelt transzformációk* elvét [J4, K21], amelyek a tranzakciók során bekövetkező modellváltozásokat perzisztáló változásmodelleket dolgozzák fel a modelltranszformációk forrás- és/vagy célmodelljeként. Az inkrementális modelltranszformációk hatékonyan felhasználhatók továbbá a tervezési tér felderítési probléma megoldására is [J7, J8, K13–K15].

3.4. Modelltranszformációk terminálásának analízise

Kutatási módszer A gráftranszformációs szabályokon alapuló modelltranszformációk formális analízise egy komplex feladat, hiszen bonyolult modellek lekérdezésének és manipulációjának helyességét kell igazolnunk. Hatékony analízis módszerek kidolgozásához elengedhetetlenek a különféle absztrakciós technikák.

A Petri háló — a közérthető grafikus jelölésrendszerének és a rendelkezésre álló analíziseszközök széles palettájának köszönhetően — gyakran használt formalizmus konku-

rens rendszerek dinamikus viselkedésének modellezésére. Így célszerűnek látszott a Petri hálók felhasználása a modelltranszformációs rendszerek terminálásának vizsgálatára.

Egy Petri hálós absztrakciónak természetesen helyesnek kell lennie abban az értelemben, hogy a gráftranszformációs rendszer minden futásához hozzá kell tudjuk rendelni a kapcsolódó Petri háló egy futását. Ugyanakkor a Petri háló egy futása az eredeti gráftranszformációs rendszer több futását is reprezentálhatja, így az absztrakciónk nem feltétlenül teljes. Ebben az esetben a Petri háló szimulálja a gráftranszformációs rendszert.

A célom az volt, hogy a Petri hálók elméletének ismert eredményeit és analízis technikáit (pl. invariánsok számítása) felhasználjam a modelltranszformációk formális helyességellenőrzésére. Az információvesztés (azaz a gráfstruktúra absztrakciója) és a terminálás algoritmikus eldönthetetlensége miatt természetesen csak részleges döntési technika kidolgozása lehetséges, azaz vagy sikerül bizonyítanunk az eredeti gráftranszformációs rendszer egy helyességi tulajdonságát, vagy bizonytalan („nem tudom”) választ kapunk.

Új tudományos eredmények A 4. tézisben a modelltranszformációk terminálásának formális igazolása céljából egy absztrakciós technikát javasoltam gráftranszformációs szabályokat Petri hálókba vetítve.

4. tézis. Kidolgoztam egy formális analízis módszert a gráftranszformációs rendszerekkel specifikált modelltranszformációk terminálásának vizsgálatára. A megközelítés először a gráftranszformációs rendszerek egy Petri hálós absztrakcióját származtatja, majd algebrai módszerek segítségével ellenőrzi a terminálás egy elégséges kritériumát [J7, J8, J20, K9, K14, K15, K27].

4/1 **Gráftranszformációs rendszerek Petri hálós absztrakciója.** Kidolgoztam a negatív alkalmazási feltételekkel rendelkező gráftranszformációs rendszerek egy (számossági) Petri hálókat származtató absztrakcióját, amely elvonatkoztat a konkrét gráfstruktúrától.

4/2 **Szimulációs bizonyítás.** Igazoltam, hogy a származtatott számossági Petri háló szimulálja az eredeti gráftranszformációs rendszert, azaz, utóbbi minden futási útjához hozzárendelhető a számossági Petri háló egy futási útja.

4/3 **Elégséges terminálási kritérium.** A Petri hálós absztrakció algebrai analízisére építve egy elégséges kritériumot definiáltam gráftranszformációs rendszerek terminálására.

Gyakorlati jelentőség Noha e tézis elsődlegesen elméleti jelentőségű, a Petri hálós absztrakciós technikát nemrégiben keresési stratégiaként is sikerrel használtuk fel a modellek fölött értelmezett tervezési tér felderítési (design space exploration) problémák megoldására [J7, J8, K14, K15]. Ugyanezen absztrakciós technika került alkalmazásra a gráftranszformációs rendszerek együttes optimalizációjára és verifikációjára [J20] továbbá futási utak visszavetítésére is [K12].

4. A tudományos eredmények hasznosítása

Végezetül szeretnék egy rövid összegzést adni a téziseimben közölt új tudományos eredmények hasznosításáról, különös tekintettel a kiemelt műszaki alkotásként is megjelölt VIATRA2 modelltranszformációs keretrendszerre, amelyet számos esetben sikerrel alkalmaztunk a szolgáltatás-orientált rendszerek és a kritikus beágyazott rendszerek területén.

4.1. A VIATRA2 modelltranszformációs keretrendszer

A VIATRA2 [4,J14] keretrendszer célja, hogy általános és magasszintű támogatást adjon a modellezési nyelveken belül és azok között definiált modelltranszformációk mérnöki fejlesztésének teljes életciklusára, beleértve a transzformációk specifikációjának, tervezésének, végrehajtásának, validálásának és karbantartásának fázisait. A nyílt forráskódú VIATRA2 keretrendszer hivatalosan is az Eclipse Generative Modeling Tools (GMT) alprojekt részét képezi.

A VIATRA2 modelltranszformációs rendszer sajátosságai a következők: (1) egy hierarchikus modelltér nagy modellek és metamodellek egységes tárolására; (2) deklaratív és imperatív elemeket is tartalmazó hibrid transzformációs nyelv; (3) egy nagy teljesítményű futtatókörnyezet, amely támogatja az inkrementális és eseményvezérelt (élő) modelltranszformációkat, ahol komplex modellváltozások automatikusan triggerelhetik a transzformációk végrehajtását, akár egymillió elemet tartalmazó modellek felett.

A VIATRA2 keretrendszert, melynek alapítója és tudományos vezetője vagyok, 2004 óta fejleszti fiatal kutatók, doktoranduszok és egyetemi hallgatók egy tehetséges és lelkes csapata a BME Méréstechnika és Információs Rendszerek Tanszékén a Hibatűrő Rendszerek Kutatócsoportban és az OptXware Kutatás-Fejlesztési Kft. munkatársaival való együttműködésben. A kiemelt kontribútorok (névsorban) a következők: Balogh András, Balogh Zoltán, Bergmann Gábor, Hegedüs Ábel, Horváth Ákos, Ökrös András, Ráth István, Schmidt András, Ujhelyi Zoltán, Vágó Dávid és Varró Gergely.

Pataricza András professzor számtalan ötlete és javaslata szintén kiemelten hasznosnak bizonyult a fejlesztés folyamán. Nagyon hálás vagyok továbbá azoknak a kutatóknak és hallgatóknak, akik a VIATRA2 keretrendszer aktív használói voltak, és értékes visszajelzéseikkel segítették a fejlesztői csapatot.

Felhasználók A VIATRA2 keretrendszer több Európai Uniós kutatási projektben szolgált modelltranszformációs technológiaként, különösen a kritikus beágyazott rendszerek (DECOS, DIANA, MOGENTES, SecureChange EU projektek) és a szolgáltatás-orientált alkalmazások (SENSORIA FP6 projekt) területén. Így e projektek akadémiai és ipari partnerei lettek a keretrendszer első (külső) felhasználói, egyúttal jelentős nemzetközi láthatóságot is biztosítva a VIATRA2 keretrendszernek. A VIATRA2 rendszer rendszeres használatáról tudunk az ARCS és TU Vienna (Ausztria), University of Leicester (Egysült Királyság), LMU München és TU Kaiserslautern (Németország), University of Trento és University of Pisa (Olaszország), University of Waterloo (Kanada) valamint a Georgia University of Technology (USA) kutatási intézményeiben.

4.2. Modelltranszformációk szolgáltatás-orientált rendszerekben

A SENSORIA EU kutatási projekt egy közérthető, modellvezérelt módszertant dolgozott ki az üzleti szolgáltatások tervezésére, amely magában foglalt (1) egy új szolgáltatás modellezési nyelvet, (2) kvantitatív és kvalitatív módszereket a szolgáltatások formális analízisére, (3) automatikus megoldásokat a szolgáltatások telepítésére, és (4) transzformációkat a régi szolgáltatások újratervezésére. A modelltranszformációk központi technológiát adtak a szolgáltatások modellvezérelt fejlesztése során azáltal, hogy áthidalták a különböző szolgáltatás-orientált nyelvek és fejlesztőeszközök közötti szakadékokat. A projekt számos eredménye épül közvetlenül a modelltranszformációk használatára:

– **BPEL folyamatok automatikus és formális analízise.** A szabványos BPEL jelölésrendszerrel leírt üzleti folyamatok [27] konzisztenciájának formális ellenőrzését a SAL modellellenőrző eszköz [2] segítségével végeztük el, amely kimerítő bejárást végez egy dinamikus viselkedésmodell összes lehetséges futási útján, hogy eldöntse a kívánt tulajdonság (követelmény) teljesülését. A SAL modelleket VIATRA2 modelltranszformációk segítségével automatikusan származtattuk [J10, K17].

– **A modellellenőrzés eredményeinek visszavetítése a BPEL folyamatokba.** A SAL modellellenőrző által visszaadott formális analízis eredményeinek visszavetítése a BPEL modellekbe egy visszafelé irányuló modelltranszformáció segítségével történt [K12], amely a két formalizmus futási útjai között létesített leképezést változásvezérelt transzformációk [K21] felhasználásával.

– **Modellvezérelt teljesítőképesség analízis.** A teljesítőképesség egy rendszerszintű nemfunkcionális paraméter, amely a hibatűrési technikák teljesítményre vetített költségét méri fel. Egy modellvezérelt teljesítőképesség analízist dolgoztunk ki [K11], amely UML alapú szolgáltatásmodellekből formális processz algebrai leírásokat származtat a PEPA [13] keretrendszer számára. A rendszerszintű formális teljesítőképesség modell az egyes komponensek teljesítőképességét leíró könyvtár alapján került származtatásra az UML komponens diagramok által vezérelten.

– **Szolgáltatások modellvezérelt telepítése.** A szolgáltatások telepítése során szükséges konfigurációs leíró fájlok automatikus generálására generikus modelltranszformációk egész láncolatát használtuk. Ennek segítségével megközelítésünket számos szabványos szolgáltatás-platformra adaptáltuk [B4, J5, K18].

E kutatásokat Gönczy Lászlóval, Hegedüs Ábellel, Kovács Mátéval, Kövi Andrással, Ráth Istvánnal, Bergmann Gáborral, Déri Zsolttal, Somodi Tiborral és Bende Tiborral együtt végeztük az én szakmai irányításom alatt.

4.3. Modelltranszformációk kritikus beágyazott rendszerekben

A VIATRA2 modelltranszformációs rendszert rendszeresen használtuk a biztonságkritikus beágyazott rendszerek tervező és verifikációs eszközeinek fejlesztésére a DECOS, DIANA, MOGENTES, SENSORIA és SecureChange Európai Uniós kutatási projektek keretében, továbbá az ipari partnerek által koordinált ARTEMIS platformhoz kötődő INDEXYS projektben.

– **Modellvezérelt eszközintegráció:** A modelltranszformációk központi szerepet játszottak a különféle ipari tervezőeszközök közötti átjárások, hidak létesítésében több eszközintegrációs probléma esetén, amelyeket egy formálisan definiált fejlesztési folyamat vezérelt [16, B1, J6].

– **Modellvezérelt tervezőeszközök:** Az interaktív, felhasználó által vezérelt modelltranszformációk adták az elméleti alapját a repülőgép- és autóipar számára készített tervezőszoftvereknek, amelyek konfigurációs táblák és allokációs leíró állományok modellvezérelt fejlesztését támogatják [B1, K16].

– **Transzformációk ontológia alapú konzisztenciaellenőrzéshez:** Szakterület-specifikus modellezőnyelvek és UML alapú tervezői modellek konzisztenciaellenőrzését egy ontológia alapú megközelítéssel végeztük. Itt magasszintű SysML és UML modellekből kiindulva modelltranszformációk automatizálták az ontológiai leírások generálását (például OWL dokumentumok formájában).

E kutatások fő stratégiáját Pataricza Andrással és Majzik Istvánnal együtt határoztuk meg, további kontribútorok (névsorban) Balogh András, Gönczy László, Horváth Ákos, Polgár Balázs, Ráth István és Varró-Gyapay Szilvia voltak.

Hivatkozások

Az értekezéshez kapcsolódó saját közlemények

— Könyvek és könyvfejezetek —

[B1] A. Balogh, G. Bergmann, G. Csertán, L. Gönczy, Á. Horváth, I. Majzik, A. Pataricza, B. Polgár, I. Ráth, D. Varró, and G. Varró. Workflow-driven tool integration using model transformations. In *Graph Transformations and Model-Driven Engineering - Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday*, volume 5765 of *LNCS*, pages 224–248. Springer, 2010.

[B2] G. Bergmann, A. Boronat, R. Heckel, P. Torrini, I. Ráth, and D. Varró. Advances in model transformation by graph transformations: Specification, analysis and execution. In M. Wirsing and M. Hölzl, editors, *Rigorous Software Engineering for Service-Oriented Systems*, volume 6582 of *LNCS*. Springer, 2011.

[B3] L. Gönczy, Á. Hegedüs, and D. Varró. Methodologies for model-driven development and deployment: an overview. In M. Wirsing and M. Hölzl, editors, *Rigorous Software Engineering for Service-Oriented Systems*, volume 6582 of *LNCS*. Springer, 2011.

[B4] L. Gönczy and D. Varró. *Design and Deployment of Service Oriented Applications with Non-Functional Requirements*, chapter Design and Deployment of Service Oriented Applications with Non-Functional Requirements, pages 315–340. IGI, New York, 2011.

[B5] L. Grunske, L. Geiger, A. Zündorf, N. Van Eetvelde, P. Van Gorp, and D. Varró. *Model Driven Software Engineering*, chapter Using Graph Transformation for Practical Model Driven Software Engineering, pages 91–118. Springer, 2005.

[B6] A. Pataricza and D. Varró. *Formal Methods in Computing*, chapter Metamodeling and Model Transformations, pages 357–425. Akadémiai Kiadó, 2005.

— Nemzetközi folyóiratban megjelent közlemények —

[J1] Z. Balogh and D. Varró. Model transformation by example using inductive logic programming. *Software and Systems Modeling*, 8(3):347–364, 2009. IF: 1.533.

[J2] G. Bergmann, Á. Horváth, I. Ráth, and D. Varró. Experimental assessment of combining pattern matching strategies with VIATRA2. *Software Tools for Technology Transfer*, 12(3-4):211–230, 2010.

[J3] G. Bergmann, I. Ráth, and D. Varró. Parallelization of graph transformation based on incremental pattern matching. *Electronic Communications of EASST*, 18, 2009.

[J4] G. Bergmann, I. Ráth, G. Varró, and D. Varró. Change-driven model transformations: Change (in) the rule to rule the change. *Software and Systems Modeling*, 2011. IF: 1.27.

[J5] S. Gilmore, L. Gönczy, N. Koch, P. Mayer, M. Tribastone, and D. Varró. Non-functional properties in the model-driven development of service-oriented systems. *Software and Systems Modeling*, 10(3):287–311, 2011. IF: 1.27.

[J6] L. Gönczy, I. Majzik, Á. Horváth, D. Varró, A. Balogh, Z. Micskei, and A. Pataricza. Tool support for engineering certifiable software. *Electr. Notes Theor. Comput. Sci.*, 238(4):79–85, 2009.

[J7] Á. Hegedüs, A. Horváth, and D. Varró. Towards guided trajectory exploration of graph transformation systems. *Electr. Comm. of the EASST*, 40:1–20, 2011.

[J8] A. Horváth and D. Varró. Dynamic constraint satisfaction problems over models. *Software and Systems Modeling*, 2011. IF: 1.27.

[J9] Á. Horváth, D. Varró, and G. Varró. Generic search plans for matching advanced graph patterns. *Electronic Communications of the EASST*, 6, 2007.

[J10] M. Kovács, L. Gönczy, and D. Varró. Formal analysis of BPEL workflows with compensation by model checking. *Int. Journal of Computer Systems and Engineering*, 23(5):35–49, 2008. IF: 0.277.

[J11] I. Ráth, A. Ökrös, and D. Varró. Synchronization of abstract and concrete syntax in domain-specific modeling languages. *Software and Systems Modeling*, 9(4):453–471, 2010. IF: 1.533.

[J12] D. Varró. Towards symbolic analysis of visual modelling languages. *Electronic Notes in Theoretical Computer Science*, 72(3):51–64, 2003.

[J13] D. Varró. Automated formal verification of visual modeling languages by model checking. *Software and Systems Modeling*, 3(2):85–113, 2004.

[J14] D. Varró and A. Balogh. The model transformation language of the VIATRA2 framework. *Science of Computer Programming*, 68(3):214–234, 2007. IF: 0.832.

[J15] G. Varró, K. Friedl, and D. Varró. Graph transformation in relational databases. *Electronic Notes in Theoretical Computer Science*, 127(1):167–180, 2005.

[J16] G. Varró, K. Friedl, and D. Varró. Implementing a graph transformation engine in relational databases. *Software and Systems Modelling*, 5(3):313–341, 2006.

[J17] G. Varró and D. Varró. Graph transformation with incremental updates. *Electronic Notes in Theoretical Computer Science*, 109:71–83, 2004.

[J18] G. Varró, D. Varró, and K. Friedl. Adaptive graph pattern matching for model transformations using model-sensitive search plans. *Electronic Notes in Theoretical Computer Science*, 152:191–205, 2006.

[J19] G. Varró, D. Varró, and A. Schürr. Incremental graph pattern matching: Data structures and initial experiments. *Electronic Communications of the EASST*, 4, 2006.

[J20] S. Varró-Gyapay and D. Varró. Optimization in graph transformation systems using Petri net based techniques. *Electronic Communications of the EASST*, 2, 2006.

— Konferencia-kiadványokban megjelent közlemények —

[K1] A. Balogh and D. Varró. Advanced model transformation language constructs in the VIATRA2 framework. In *Proc. ACM Symposium on Applied Computing (SAC 2006)*, pages 1280–1287. ACM Press, 2006. Acc. rate = 32%.

[K2] A. Balogh, G. Varró, D. Varró, and A. Pataricza. Compiling model transformations to EJB3-specific transformer plugins. In *Proc. ACM Symposium on Applied Computing (SAC 2006)*, pages 1288–1295. ACM Press, 2006. Acc. rate = 32%.

[K3] G. Bergmann, Á. Horváth, I. Ráth, and D. Varró. A benchmark evaluation of incremental pattern matching in graph transformation. In *Proc. 4th Int. Conf. on Graph Transformations, ICGT 2008*, volume 5214 of *LNCS*, pages 396–410. Springer, 2008. Acc. rate = $\tilde{4}0\%$.

[K4] G. Bergmann, Á. Horváth, I. Ráth, and D. Varró. Efficient model transformations by combining pattern matching strategies. In *Proc. Theory and Practice of Model Transformations, Second Int. Conf., ICMT 2009.*, volume 5563 of *LNCS*, pages 20–34. Springer, 2009. Acc. rate = 22%.

[K5] G. Bergmann, Á. Horváth, I. Ráth, and D. Varró. Incremental evaluation of model queries over EMF models: A tutorial on EMF-IncQuery. In *Proc. ECMFA 2011: Modelling Foundations and Applications - 7th European Conference*, volume 6698 of *LNCS*, pages 389–390. Springer, 2011.

[K6] G. Bergmann, Á. Horváth, I. Ráth, D. Varró, A. Balogh, Z. Balogh, and A. Ökrös. Incremental evaluation of model queries over EMF models. In *Model Driven Engineering Languages and Systems - 13th Int. Conf., MODELS 2010, Proceedings, Part I*, volume 6394 of *LNCS*, pages 76–90. Springer, 2010. Acc. rate: 21%.

[K7] G. Bergmann, A. Ökrös, I. Ráth, D. Varró, and G. Varró. Incremental pattern matching in the VIATRA model transformation system. In *Proc. Graph and Model Transformations (GRAMOT 2008)*. ACM, 2008.

[K8] G. Bergmann, Z. Ujhelyi, I. Ráth, and D. Varró. A graph query language for EMF models. In *Proc. Int. Conf. on Model Transformation*, volume 6707 of *LNCS*, pages 167–182. Springer, 2011. Acc. rate = 27%.

[K9] H. Ehrig, K. Ehrig, J. de Lara, G. Taentzer, D. Varró, and S. Varró-Gyapay. Termination criteria for model transformation. In M. Cerioli, editor, *Proc. FASE 2005: Int. Conf. on Fundamental Approaches to Software Engineering*, volume 3442 of *LNCS*, pages 49–63, Edinburgh, UK,, April 2005. Springer. Acc. rate = 22%.

[K10] K. Ehrig, E. Guerra, J. de Lara, L. Lengyel, T. Levendovszky, U. Prange, G. Taentzer, D. Varró, and S. Varró-Gyapay. Model transformation by graph transformation: A comparative study. In *MTiP 2005, Int. Workshop on Model Transformations in Practice (Satellite Event of MoDELS 2005)*, 2005.

[K11] L. Gönczy, Z. Déri, and D. Varró. Model transformations for performability analysis of service configurations. In *Models in Software Engineering, Workshops and Symposia at MODELS 2008. Reports and Revised Selected Papers*, volume 5421 of *LNCS*, pages 153–166. Springer, 2008.

[K12] Á. Hegedüs, G. Bergmann, I. Ráth, and D. Varró. Back-annotation of simulation traces with change-driven model transformations. In *Proceedings of the Eighth Int. Conf. on Software Engineering and Formal Methods (SEFM 2010)*, pages 145–155. IEEE Computer Society, 2010. Acc. rate: 22%.

[K13] Á. Hegedüs, Á. Horváth, M. C. Branco, I. Ráth, and D. Varró. Quick fix generation for DSMLs. In *Proc. VL/HCC 2011: IEEE Symposium on Visual Languages and Human-Centric Computing,*, pages 17–24. IEEE, 2011. Acc. rate = 33%.

[K14] Á. Hegedüs, A. Horváth, and D. Varró. A model-driven framework for guided design space exploration. In *Proc. ASE 2011: 26th IEEE/ACM Int. Conf. On Automated Software Engineering*, pages 173–182. IEEE Computer Society, 2011. Acc. rate = 16%, **ACM Distinguished Paper Award**.

[K15] Á. Horváth and D. Varró. CSP(M): Constraint Satisfaction Problem over Models. In *Proc. Model Driven Engineering Languages and Systems, 12th Int. Conf., MODELS 2009*, volume 5795 of *LNCS*, pages 107–121. Springer, 2009. Acc. rate: 18%.

[K16] Á. Horváth, D. Varró, and T. Schoofs. Model-driven development of ARINC 653 configuration tables. In *29th IEEE & AIAA Digital Avionics System Conference (DASC)*, pages 5.A.5–1 – 5.A.5–115. IEEE, 2010.

[K17] M. Kovács, L. Gönczy, and D. Varró. Formal modeling of BPEL workflows including fault and compensation handling. In *EFTS '07: Proceedings of the 2007 Workshop on Engineering Fault Tolerant Systems*, page 1. ACM, 2007.

[K18] A. Kövi and D. Varró. An Eclipse-based framework for AIS service configurations. In *Proc. 4th Int. Service Availability Symposium, ISAS 2007*, volume 4526 of *LNCS*, pages 110–126. Springer, 2007.

[K19] I. Ráth, G. Bergmann, A. Ökrös, and D. Varró. Live model transformations driven by incremental pattern matching. In *Proc. First Int. Conf. on the Theory and Practice of Model Transformations (ICMT 2008)*, volume 5063 of *LNCS*, pages 107–121. Springer, 2008. Acc. rate = 31%.

[K20] I. Ráth, D. Vago, and D. Varró. Design-time simulation of domain-specific models by incremental pattern matching. In *IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2008, Proceedings*, pages 219–222. IEEE, 2008. Acc. rate = 29%.

[K21] I. Ráth, G. Varró, and D. Varró. Change-driven model transformations. In *Model Driven Engineering Languages and Systems, 12th Int. Conf., MODELS 2009. Proceedings*, volume 5795 of *LNCS*, pages 342–356. Springer, 2009. Acc. rate: 18%, **Springer Best Paper Award and ACM Distinguished Paper Award**.

[K22] A. Rensink, Á. Schmidt, and D. Varró. Model checking graph transformations: A comparison of two approaches. In *Proc. ICGT 2004: Second Int. Conf. on Graph Transformation*, volume 3256 of *LNCS*, pages 226–241. Springer, 2004.

[K23] D. Varró. Model transformation by example. In *Proc. Model Driven Engineering Languages and Systems (MODELS 2006)*, volume 4199 of *LNCS*, pages 410–424. Springer, 2006. Acc. rate = 29%.

[K24] D. Varró and Z. Balogh. Automating model transformation by example using inductive logic programming. In *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC 2007)*, pages 978–984. ACM Press, 2007. Acc. rate = 32%.

[K25] D. Varró and A. Pataricza. Automated formal verification of model transformations. In *CSDUML 2003: Critical Systems Development in UML; Proceedings of the UML'03 Workshop*, number TUM-I0323 in Technical Report, pages 63–78. Technische Universität München, September 2003.

[K26] D. Varró and A. Pataricza. Generic and meta-transformations for model transformation engineering. In *Proc. UML 2004: 7th Int. Conf. on the Unified Modeling Language*, volume 3273 of *LNCS*, pages 290–304. Springer, 2004. Acc. rate = 22%.

[K27] D. Varró, S. Varró-Gyapay, H. Ehrig, U. Prange, and G. Taentzer. Termination analysis of model transformations by Petri nets. In *Proc. Third Int. Conf. on Graph Transformation (ICGT 2006)*, volume 4178 of *LNCS*, pages 260–274. Springer, 2006. Acc. rate = 45%.

[K28] G. Varró, Á. Horváth, and D. Varró. Recursive graph pattern matching: With magic sets and global search plans. In *Proc. Third Int. Workshop and Symposium on Applications of Graph Transormation with Industrial Relevance (AGTIVE 2007)*, volume 5088 of *LNCS*. Springer, 2008.

További hivatkozások

[1] M. Asztalos, L. Lengyel, and T. Levendovszky. Towards automated, formal verification of model transformations. In *Proc. 3rd Int. Conf. on Software Testing, Verification and Validation (ICST 2010)*, pages 15–24. IEEE Computer Society, 2010.

[2] S. Bensalem, V. Ganesh, Y. Lakhnech, C. Munoz, S. Owre, H. Rueß, J. Rushby, V. Rusu, H. Saïdi, N. Shankar, E. Singerman, and A. Tiwari. An overview of SAL. In C. M. Holloway, editor, *LFM 2000: Fifth NASA Langley Formal Methods Workshop*, pages 187–196, 2000.

[3] A. Bondavalli, M. Dal Cin, D. Latella, I. Majzik, A. Pataricza, and G. Savoia. Dependability analysis in the early phases of UML based system design. *Inter. Journal of Computer Systems - Science & Engineering*, 16(5):265–275, 2001.

[4] Budapest University of Technology and Economics. *VIATRA2 Model Transformation Framework*. `http://www.eclipse.org/gmt/VIATRA2`.

[5] S. Burmester, H. Giese, M. Hirsch, D. Schilling, and M. Tichy. The Fujaba Real-Time tool suite: model-driven development of safety-critical, real-time systems. In *27th Int. Conf. on Software Engineering (ICSE 2005), 15-21 May 2005, St. Louis, Missouri, USA*, pages 670–671. ACM, 2005.

[6] K. Chen, J. Sztipanovits, and S. Neema. Toward a semantic anchoring infrastructure for domain-specific modeling languages. In *EMSOFT '05: Proceedings of the 5th ACM Int. Conf. on Embedded Software*, pages 35–43, New York, NY, USA, 2005. ACM. ISBN 1-59593-091-4.

[7] A. Cypher, editor. *Watch What I Do: Programming by Demonstration*. The MIT Press, 1993.

[8] M. Didonet Del Fabro and P. Valduriez. Towards the efficient development of model transformations using model weaving and matching transformations. *Software and System Modeling*, 8(3):305–324, 2009.

[9] H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors. *Handbook on Graph Grammars and Computing by Graph Transformation*, volume 2: Applications, Languages and Tools. World Scientific, 1999.

[10] M. Erwig. Toward the automatic derivation of XML transformations. In *1st Int. Workshop on XML Schema and Data Management (XSDM'03)*, volume 2814 of *LNCS*, pages 342–354. Springer, 2003.

[11] J.-R. Falleri, M. Huchard, M. Lafourcade, and C. Nebut. Metamodel matching for automatic model transformation generation. In *Model Driven Engineering Languages and Systems, 11th Int. Conf., MoDELS 2008*, volume 5301 of *LNCS*, pages 326–340. Springer, 2008.

[12] I. García-Magariño, J. J. Gómez-Sanz, and R. Fuentes-Fernández. Model transformation by-example: An algorithm for generating many-to-many transformation rules in several model transformation languages. In *ICMT '09: Proceedings of the 2nd Int. Conf. on Theory and Practice of Model Transformations*, pages 52–66. Springer, 2009. ISBN 978-3-642-02407-8.

[13] S. Gilmore and J. Hillston. The PEPA workbench: A tool to support a process algebra-based approach to performance modelling. In *Proc. of 7th Intern. Conf. on Computer Performance Evaluation, Modeling Techniques and Tools,*, volume 794 of *LNCS*, pages 353–368. Springer, 1994.

[14] J. Greenyer and E. Kindler. Comparing relational model transformation technologies: implementing query/view/transformation with triple graph grammars. *Software and System Modeling*, 9(1):21–46, 2010.

[15] Y. Gurevich. The sequential ASM thesis. *Bulletin of the European Association for Theoretical Computer Science*, 67:93–124, 1999.

[16] W. Herzner, B. H. G. Csertán, and A. Balogh. The DECOS tool-chain: Model-based development of distibuted embedded safety-critical real-time systems. In *Proc. of the DECOS/ERCIM Workshop at SAFECOMP 2006*, pages 22–24. ERCIM, 2006.

[17] S. Islam, N. Suri, A. Balogh, G. Csertán, and A. Pataricza. An optimization based design for integrated dependable real-time embedded systems. *Design Autom. for Emb. Sys.*, 13(4):245–285, 2009.

[18] F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev. ATL: A model transformation tool. *Sci. Comput. Program.*, 72(1-2):31–39, 2008.

[19] G. Karsai and A. Narayanan. On the correctness of model transformations in the development of embedded systems. In *Composition of Embedded Systems. Scientific and Industrial Issues, 13th Monterey Workshop 2006, Revised Selected Papers*, volume 4888 of *LNCS*, pages 1–18. Springer, 2007.

[20] M. Kessentini, H. A. Sahraoui, and M. Boukadoum. Model transformation as an optimization problem. In *Model Driven Engineering Languages and Systems, 11th Int. Conf., MoDELS 2008*, volume 5301 of *LNCS*, pages 159–173. Springer, 2008.

[21] A. Kleppe, J. Warmer, and W. Bast. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley, 2003.

[22] S. Lechner and M. Schrefl. Defining web schema transformers by example. In V. Marik, W. Retschitzegger, and O. Stepankova, editors, *DEXA*, volume 2736 of *LNCS*, pages 46–56. Springer, 2003.

[23] L. Lengyel, T. Levendovszky, and H. Charaf. Validated model transformation-driven software development. *IJCAT*, 31(1/2):106–119, 2008.

[24] Z. Á. Mann, A. Orbán, and P. Arató. Finding optimal hardware/software partitions. *Formal Methods in System Design*, 31:241–273, 2007.

[25] S. P. Miller, A. C. Tribble, M. W. Whalen, and M. P. E. Heimdahl. Proving the shalls. *STTT*, 8(4-5):303–319, 2006.

[26] S. Muggleton and L. de Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19-20:629–679, 1994.

[27] OASIS. Web Services Business Process Execution Language Version 2.0 (OASIS Standard), 2007.

[28] Object Management Group. QVT: MOF 2.0 Query / View / Transformation, 2008. `http://www.omg.org/spec/QVT/1.0/`.

[29] K. Ono, T. Koyanagi, M. Abe, and M. Hori. XSLT stylesheet generation by example with WYSIWYG editing. In *Proceedings of the 2002 Symposium on Applications and the Internet (SAINT 2002)*, pages 150–161. IEEE Computer Society, 2002.

[30] A. Pataricza. Model-based dependability analysis, 2008. DSc thesis, Hungarian Academy of Sciences.

[31] D. Plump. Termination of graph rewriting is undecidable. *Fundam. Inform.*, 33(2): 201–209, 1998.

[32] A. Repenning and C. Perrone. Programming by example: programming by analogous examples. *Communications of the ACM*, 43(3):90–97, 2000.

[33] R. Robbes and M. Lanza. Example-based program transformation. In *Model Driven Engineering Languages and Systems, 11th Int. Conf., MoDELS 2008. Proceedings*, volume 5301 of *LNCS*, pages 174–188. Springer, 2008.

[34] E. Schoitsch, E. Althammer, H. Eriksson, J. Vinter, L. Gönczy, A. Pataricza, and G. Csertán. Validation and certification of safety-critical embedded systems - the decos test bench. In *Computer Safety, Reliability, and Security, 25th Int. Conf., SAFECOMP 2006*, volume 4166 of *LNCS*, pages 372–385. Springer, 2006.

[35] A. Schürr. Specification of graph translators with triple graph grammars. In B. Tinhofer, editor, *Proc. WG94: International Workshop on Graph-Theoretic Concepts in Computer Science*, number 903 in LNCS, pages 151–163. Springer, 1994.

[36] G. Simon, G. Karsai, G. Biswas, S. Abdelwahed, N. Mahadevan, T. Szemethy, G. Péceli, and T. Kovácsházy. Model-based fault-adaptive control of complex dynamic systems. In *Proceedings of the 20th IEEE Instrumentation and Measurement Technology Conference, IMTC/2003*, pages 176–181. IEEE, 2003.

[37] M. Strommer, M. Murzek, and M. Wimmer. Applying Model Transformation By-Example on Business Process Modeling languages. In *Proc. 3rd Int. Workshop on Foundations and Practices of UML (ER 2007)*, volume 4802 of *LNCS*, pages 116–125. Springer, 2007.

[38] Y. Sun, J. White, and J. Gray. Model transformation by demonstration. In *MODELS '09: Proceedings of the 12th Int. Conf. on Model Driven Engineering Languages and Systems*, pages 712–726. Springer, 2009. ISBN 978-3-642-04424-3.

[39] J. Sztipanovits and G. Karsai. Model-integrated computing. *IEEE Computer*, 30(4): 110–111, 1997.

[40] L. L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. Data-driven understanding and refinement of schema mappings. In *Proc. ACM SIGMOD Conf. on Management of Data*, pages 485–496, 2001.

[41] M. M. Zloof. Query-by-example: the invocation and definition of tables and forms. In D. S. Kerr, editor, *VLDB*, pages 1–24. ACM, 1975.